

In the Claims

This listing of claims will replace all prior versions and listings of claims in the application:

1        1. (Previously Presented) A method of performing a Fast  
2 Fourier Transform in a data processing apparatus having a data  
3 cache smaller than the data set of the Fast Fourier Transform,  
4 comprising the steps of:

5                dividing said input data into R continuous data sets where  
6 each of said R continuous data sets fit within the data cache;

7                disposing said input data into memory, each R continuous data  
8 set in continuous memory locations with a space in memory locations  
9 from an end of one continuous data set to a beginning of a next  
10 continuous data set equal to the size of a cache line;

11                separately and independently performing a first stage radix-R  
12 butterfly computations on all the the R continuous data sets  
13 thereby producing R independent intermediate data sets each of  
14 which fits within the data cache; and

15                successively performing second and all subsequent stage  
16 butterfly computations on each independent intermediate data set in  
17 turn producing corresponding output data.

2. (Canceled)

1        3. (Original) The method of claim 1, wherein:  
2                said radix-R is radix-2.

1        4. (Original) The method of claim 1, wherein:  
2                said radix-R is radix-4.

5. (Canceled)

1        6. (Previously Presented) The method of performing an  
2 N-point radix-R Fast Fourier Transform in a data processing  
3 apparatus having a data cache comprising the steps of:  
4        comparing the data set of input data and twiddle factors with  
5 the size of the data cache;  
6        if said data set is smaller than said data cache, performing  
7 said Fast Fourier Transform in  $\log_R N$  stages on all the data set in  
8 one pass; and  
9        if said data set is larger than said data cache but smaller  
10 than R times the data cache  
11                dividing said input data into R continuous data sets  
12 where each of said R continuous data sets fit within the data  
13 cache;  
14                disposing said input data into memory, each R continuous  
15 data set in continuous memory locations with a space in memory  
16 locations from an end of one continuous data set to a  
17 beginning of a next continuous data set equal to the size of a  
18 cache line;  
19                separately and independently performing a first stage  
20 radix-R butterfly computations on all the the R continuous  
21 data sets thereby producing R independent intermediate data  
22 sets in a first pass each of which fits within the data cache;  
23 and  
24                successively performing second and all subsequent stage  
25 butterfly computations on each independent intermediate data  
26 set in turn producing corresponding output data in second  
27 passes.

1        7. (Original) The method of claim 6, wherein:  
2        said Fast Fourier Transform uses complex input data and  
3 complex twiddle factors of M bytes each; and

4        said step of comparing the data set with the size of the data  
5    cache compares the data cache size to  $4 \text{ NxM}$  bytes.

1        8. (Original) The method of claim 6, wherein:  
2    said radix-R is radix-2.

1        9. (Original) The method of claim 6, wherein:  
2    said radix-R is radix-4.

3        10. (Canceled)

1        11. (Original) The method of claim 6, further comprising:  
2    if said data set is larger than R times the data cache  
3        performing I initial stages of radix-R butterfly  
4        computations on all the input data producing R independent  
5        intermediate data sets, where I is the next integer greater  
6        than  $\log_R(D/C)$ , D is the size of the data set and C is the  
7        size of the cache; and

8        successively performing all subsequent stage butterfly  
9        computations on each independent intermediate data set in turn  
10      producing corresponding output data in second passes.